Mobile App Engineering & User Experience: Project Report for "Tetwis"

Group Members:

Allen Tung, Fahad Mohammad, Jonathan Zelaya, Prit Modi

Group Contributions

Allen Tung - Project Planning and design, UI on the Main menu, taking care of problems with Git, planning, block textures, editing physics textures, adjusting spawning behavior, UI functionality, General layout, Presentation, Art asset design, Report, Presentation
Fahad Mohammad - UI on the Main menu, making backgrounds, implemented functionality to spawn random background, adding sounds/music, planning. Project design. Time attack game mode and essential files and assets related to Time attack game mode. Functionality of the timer. Presentation slides, Report.
Jonathan Zelaya - Created the physics of main aspect of game (tetromino.cs), added code to spawn pieces, move camera up (CameraController.cs), adjust pieces to reduce effect of bounce, added some block assets/backgrounds, ported game to android (switch input from keyboard to touch), explanation of game development in report, slides, project planning
Prit Modi - Project planning, making music for different parts of the game, testing and troubleshooting

Project Structure created in Unity

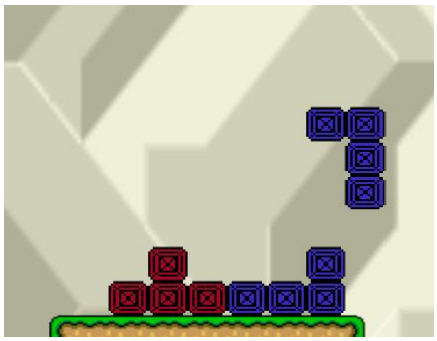
Main classes:

- Tetromino.cs
  - Controls the block physics and spawning
- Game.cs
  - Controls the general game flow, game characteristics (points, game end, etc)
- CameraController.cs
  - Controls the camera panning
- ChangeScene.cs
  - Manages the scene switching, implemented with buttons.
- Background.cs
  - Manages the background image switching
- Timer.cs
  - Implements a timer for Time mode
- Endlessgame.cs
  - Controls game flow of time mode
- Tetromino_endless.cs
  - For timed mode. Changes tetromino behavior in response to time

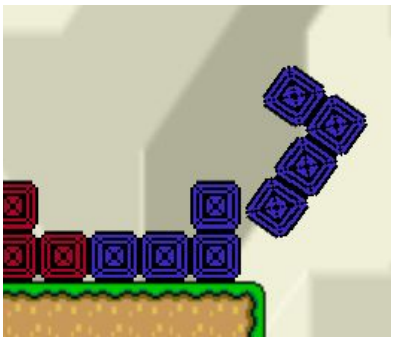
How the game works (behind the scenes)

There are many aspects to this game that had to be considered. The biggest aspect is the physics in the game. Unity can handle most of the calculations and reactions to collisions by itself with no need for user intervention. It is therefore easy to create a realistic game with not too much code.

One of the biggest problems (that was partially solved) was the effect of falling blocks colliding with stationary blocks. Whenever they landed on stationary blocks, the effect of gravity applied a collision force and shifted the piece slightly. Even the slightest touch will shift the blocks and thus lead to unexpected results. See pictures below.

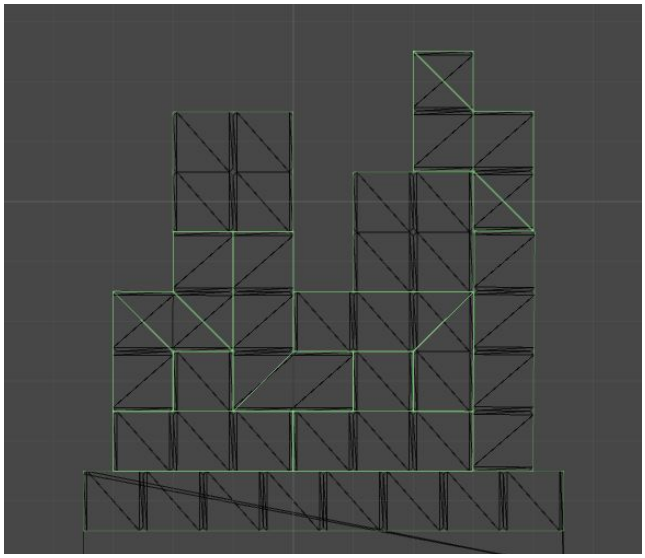


Everything going to accord to plan!



Nope!

Every piece is really just an object called a "Rigidbody 2D" (rb2d) with an associated "Collider 2D" component. This allows it to behave like any real life object. These can be moved by other objects, forces, etc. When one rb2d falls onto another one, there is a slight bounce due to physics causing undesired results. In order to create a tetris-like behavior, we would need to have zero force applied as it falls. To (partially) solve this, I decided to turn off gravity for blocks until they collided with another piece. Another way to help alleviate this problem is to increase the mass to maximum after it falls; this reduces the effect from lighter blocks falling onto it.



The picture to the left shows how Unity represents objects. The green borders are the "Collision 2D" components and are the main factors that affect collision handling. Because there is such a small margin between borders (if any), any small disturbance in the position of a block will cause unexpected collisions.

The project was implemented in Unity, because Unity has much more robust support for physics engines built into the program itself. Graphics assets are also handled more easily, with strong support for animations and transitions.

Initially, there was intended to be multiplayer support if time permitted, but heavy course loads meant that we were only able to implement a single person game mode.

The app has a homepage, with several buttons on it that take you to different game modes. There are two game modes, one for play where the limit is a physical height, and one for a timed play. In each of these modes, blocks will be spawned at the top of the screen, fall down to a fixed platform with or without user input, and then another block will be spawned. Any blocks that fall below the platform will be deleted. In each game mode, there is the option to simply return to the main menu, and select another game mode.
Once the condition for ending the game has been met, the scene will switch to the 'Game Over' scene, where there is a friendly image, and a button to return the user to the home page.

For each of these game modes, we record the user's highest lifetime score, which is the height of the tallest tower they have managed to build.

In the first game mode, 「Standard」 the user has a fixed number of lives, and they will lose one life for every block that falls off the edge of the screen. Once the user loses all their lives, the height of the tower will be computed as their score.

In the second game mode, 「Time Attack」 the user will have a limited time to place blocks. Once the time is up, their score will be computed based on the height of the tallest block currently placed. The Highscore will be saved and displayed on the main menu.